International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 15, No.1 (Mar-24)

http://dx.doi.org/10.12785/ijcds/1501103

Analyzing the Impact of Discretization Techniques on Real Time Simulation of DC Servo Motor Using FPGA

Mini K. Namboothiripad[1]

[1]Department of Electrical Engineering,
[1]Agnel Charities' Fr. C. Rodrigues Institute of Technology, Vashi
[1] Navi-Mumbai, Maharashtra, India.

Received 21 Jan. 2024, Revised 06 Mar. 2024, Accepted 12 Mar. 2024, Published 15 Mar. 2024

Abstract: This paper explains the strategies to create a hardware-based real time model of DC servomotor by utilizing the FPGA technology that can accurately simulate the behavior of the servomotor in real-time. Such FPGA based hardware model is useful for testing control algorithms, validating designs, and optimizing performance for various applications because of its reconfiguration capabilities. The uniqueness of our paper lies in the application of *C* language for modeling a DC servomotor by discretizing the model with both Backward Euler (BE) and Trapezoidal (TRZ) methods. The discretized models are then implemented on FPGA using Vivado HLS and the performance is analyzed with change in step-size by comparing with the transfer function (TF) model. With 100 µsec step-size, TRZ response is found to be matching with the TF model, however, a step-size of 0.6 µsec was required for the BE. Also analyzed the feasibility of Hardware-in-the-Loop (HIL) technique by connecting the DC servomotor in loop with the PID controller on FPGA, again by varying the step-size. Both the BE and TRZ models could track the reference speed within 2 msec, because of the PID controller, however faster dynamics was observed in case of TRZ as compared to BE, especially with larger step-size. This analysis demonstrates the impact of step-size and discretization technique on real-time modeling. By selecting suitable values, the FPGA model can be efficiently harnessed to develop a tailored control algorithm and optimize performance for diverse applications.

Keywords: Backward Euler, DC Servomotor, FPGA, High Level Synthesis, Discrete PID Controller, Real-Time Simulation, Trapezoidal, Vivado

1. Introduction

DC servo motors are versatile devices widely used in various applications across a wide range of industries and technologies [1]. Its crucial role in robotic systems, computer numerical control machines, medical devices, aerospace applications, solar tracking systems, automotive systems, etc. highlights its versatility and precision control capabilities. Its various applications in electric vehicles such as power steering, power brake, power windows, power seating, cooling fans, heating, ventilation, and air conditioning systems, etc. contribute to improved energy efficiency, enhanced vehicle control, and increased overall performance. As technology continues to advance, the role of these motors in various applications may expand even further.

Research on DC servomotors spans various aspects, including control algorithms, modeling and simulation, real-time implementations, hardware implementations, and optimization techniques, with applications in different fields. Real-time simulation of DC servomotors is a crucial step in the development and deployment of control systems. It allows rapid control prototyping and testing, hardware in the loop and software in the loop testing, etc. without the need for complete physical hardware setups [2]-[4]. It enables thorough testing, analysis, and optimization of control algorithms, ultimately leading to more reliable and efficient real-world implementations.

Simulating a DC servomotor in real-time involves using software tools and models to replicate the behaviour of the motor and its control system. Various papers are available in the literature, with different software tools, to investigate the impact of the control algorithms to enhance the precision and responsiveness of DC servomotors. A LabVIEW-based PID controller for the position control of a DC servomotor is explained in detail in paper [5]. A similar approach with LabVIEW, but a fuzzy-PI controller is developed in the paper [6] to control the speed of the DC servomotor.

The paper in [7] presents a MATLAB-based simulation model for the DC servomotor and PID controller, to enhance the performance of the motor. Paper [8] details the application of such MATLAB-based motor with PID controller in the context of a humanoid robotic arm whereas [9] explains the modeling and control of DC servomotor for the electric wheel chair arrangement. Such PID controller can be tuned with various heuristic algrothms using MATLAB such as genetic algorithm [10], [11], Krill herd algorithm [12] and Smell agent optimization algorithm [13] for the position and speed control of the DC servomotor. Whereas, the papers [14]-[16] explain the advantages of other control techniques such as fuzzy logic [14], ANN [15], and a hybrid fuzzy and position-velocity controller [16] for the speed control of DC servomotor, using MATLAB /Simulink models. In paper [17] the parameter estimation of DC servomotor is analyzed using various toolboxes in MATLAB such as the Parameter Estimation Toolbox, and System Identification Toolbox.

While LabVIEW [5], [6], and MATLAB/Simulink [7]-[17] models are powerful tools for designing and simulating control algorithms, transitioning to *C* code is often necessary for real-world deployment on embedded systems or microcontrollers [18],[19]. MATLAB coder tools are available in MATLAB to transform the designed algorithm into *C* code to make it run on embedded systems [20]. Using high level synthesis (HLS) tools, *C* code can be converted to hardware descriptive language (HDL) which is used for the implementation on Field Programmable Gate Array (FPGA). This paper explores the real-time modeling of DC Servomotors on FPGA, where the motor's characterization is created using *C* code. The transition to HDL is accomplished through the application of the Vivado HLS tool, ensuring efficient implementation on the FPGA platform.

FPGAs provide hardware-level parallelism and can execute real-time simulations of systems and control algorithms with minimal latency [21],[22]. This is crucial in real-time applications where rapid response times are essential [23-26], such as robotics and automation. Prototyping systems using FPGA can be a cost-effective solution compared to designing custom ASICs (Application-Specific Integrated Circuits). It offers a balance between performance and cost, as compared to ASICs, making them suitable for iterative development [27].

DC servomotors often require integration with sensors for feedback on position, speed, or torque. FPGA-based prototypes facilitate the integration of these sensors in real-time, helping to validate and refine the feedback control loop. Thus well-suited for hardware-in-the-loop (HIL) testing [28], [29] where the DC servomotor in FPGA can be connected to a physical or a simulated control system. It is valuable for educational purposes, providing hands-on experience in control system design.

Such prototyping also allows engineers and researchers to develop and optimize control algorithms for the DC servomotor, with different specifications, in a real-time environment. This is feasible because FPGAs are versatile and can be easily reprogrammable, hence the same FPGA platform can be repurposed to adapt to different DC servomotor applications such as control of robotic arm, conveyor system, or any other automated process. While FPGA-based implementations offer these advantages, in general, it requires expertise in hardware design and FPGA programming. However, HLS tools will facilitate the coding with high level languages such as *C* [30]-[32].

Therefore, the real-time modeling of a DC servomotor on an FPGA facilitates the development of a prototype that seamlessly integrates into control loops for continuous advancements. This real-time modeling on an FPGA is accomplished by transforming the continuous system into its discretized form, achieved through the implementation of *C* code. There are various discretization methods available in the literature [33]- [35] such as the Backward Euler (BE) method, Trapezoidal (TRZ) method, etc. Ultimately, it's crucial to evaluate the performance of available models in the specific application context to determine which discretization method better meets the accuracy and stability requirements of the system. The paper [36] explains in detail the discretization issue with the BE method for the real-time simulation of the induction motor, especially with the steady state error.

TRZ method is a second order approximation thus can be computationally more efficient and accurate than BE, especially for systems where the solution exhibits rapid changes [37]. In the case of DC servo motors, because of its stiff dynamics, the solution changes rapidly [1], thus TRZ method may provide more accurate solutions, which need to be verified.

This paper explains the real-time modeling of DC Servomotor using both BE and TRZ methods on FPGA and compares their performance with the transfer function (TF) model. The motor is modeled using 'C' code and converted to HDL using the Vivado HLS tool for its implementation on FPGA [38]. It is observed that even with a step-size of 100 μsec, the response from TRZ is found to be at par with the TF model. However, the response from BE is equivalent to that only by reducing the step-size to 6μsec. Also, analyzed the performance of these discrete time models on FPGA by connecting it in a loop with the discrete PID controller [39] for speed control. In this case also, observed a faster dynamic with the TRZ method as compared to the BE method thus verifying the superiority of TRZ.

The novelty of our paper lies in its distinctive approach of utilizing *C* for modeling a DC servomotor, incorporating both BE and TRZ discretization techniques. This model is subsequently translated into FPGA through Vivado HLS. The resulting FPGA prototypes seamlessly integrate into loops with controllers, facilitating the efficient development of advanced control algorithms customized for diverse applications. Additionally, we evaluate the feasibility of this

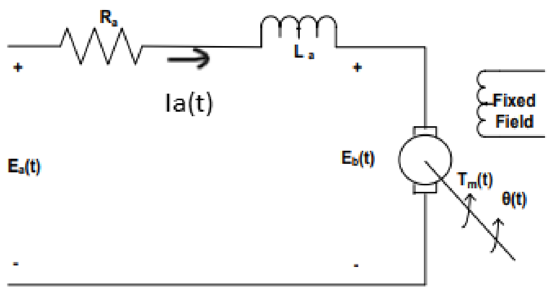

Figure 1. Equivalent circuit of a DC servomotor

Hardware-in-the-Loop (HIL) technique by incorporating it into a loop with PID implementation on FPGA. The key contribution of this paper thus lies in its advancement in leveraging HIL techniques for the design and development of sophisticated control algorithms.

2. Modelling of DC Servo Motor

The equivalent circuit of a DC servomotor can be represented as shown in Fig. 1. Using Kirchhoff's Voltage Law on this circuit, the mathematical equation can be written as,

$$R_a i_a(t) + L_a\frac{di_a(t)}{dt} + E_b(t) = E_a(t) \tag{1}$$

Where R_a and L_a are the resistance and inductance of the armature circuit, E_a is the applied DC voltage, E_b is the generated back emf and i_a is the current flowing through the armature circuit.

The back emf E_b can be written in terms of angular velocity w_m and back emf constant, K_b as,

$$E_b(t) = K_b w_m(t) \tag{2}$$

Equations 1 and 2 can be combined as,

$$R_a i_a(t) + L_a\frac{di_a(t)}{dt} + K_b w_m(t) = E_a(t) \tag{3}$$

Torque developed by the motor, T_m, can be written in terms of Moment of Inertia, J_m, and Frictional constant,D_m, of motor as,

$$T_m(t) = J_m\frac{dw_m(t)}{dt} + D_m w_m(t) \tag{4}$$

Also, the torque can be expressed in terms of armature current i_a and Torque constant K_t as,

$$T_m(t) = K_t i_a(t) \tag{5}$$

Equation 6 can be derived by combining 4 and 5,

$$K_t i_a(t) = J_m\frac{dw_m(t)}{dt} + D_m w_m(t) \tag{6}$$

And then rearranging 3 and 6 the following 7 and 8 can be derived.

$$\frac{di_a(t)}{dt} = -\frac{R_a}{L_a}i_a(t) - \frac{K_b}{L_a}w_m(t) + \frac{1}{L_a}E_a(t) \tag{7}$$

$$\frac{dw_m(t)}{dt} = \frac{K_t}{J_m}i_a(t) - \frac{D_m}{J_m}w_m(t) \tag{8}$$

Equations 7 and 8 can be discretized using a numerical method to simulate the dynamic behavior of the DC servo motor. It's important to note that the accuracy of the simulation depends on the chosen step-size and the appropriateness of the numerical method for the specific dynamics of the system.

For the TF model, assuming zero initial conditions, 3 and 6 can be written using Laplace transform as follows:

$$R_a I_a(s) + L_a sIa(s) + K_b W_m(s) = E_a(s) \tag{9}$$

$$K_t I_a(s) = J_m sW_m(s) + D_m W_m(s) \tag{10}$$

Substituting $I_a(s)$ from 10 to 9, and after the simplification, the TF can be expressed as,

$$G(s) = \frac{W_m(s)}{E_a(s)} = \frac{K_t}{J_m L_a s^2 + (R_a J_m + LaD_m)s + (R_a D_m + K_b K_t)} \tag{11}$$

A. Discretization Methods

During transient analysis, the total time interval of interest is discretized into small step-size, h, and the circuit is solved at each time-step or sample. An ordinary differential equation, $\frac{dx}{dt}$, which exists due to the presence of energy storage elements, can be solved numerically by representing it as, $slope = \frac{x_k - x_{k-1}}{t_k - t_{k-1}}$. Here, k is the current sample and $k-1$ is the previous sample, $t_k - t_{k-1}$is the step-size, h. However, this expression may be representing the slope at the samples k, $k-1$, etc. Depending upon that, various discretization techniques like BE and TRZ methods are defined.

In the BE method, $\frac{x_k - x_{k-1}}{h}$ is the slope at k. Thus $\frac{dx}{dt} = f(x)$ can be written in discrete form as, $\frac{x_k - x_{k-1}}{h} = f(x_k)$. However, in the TRZ method, $\frac{dx}{dt} = \frac{1}{2}(f(x_k) + f(x_{k-1}))$ and thus, $\frac{x_k - x_{k-1}}{h} = \frac{1}{2}(f(x_k) + f(x_{k-1}))$ is representing the differential equation.

The choice between the TRZ and BE method depends on factors such as system dynamics, stability requirements, and computational efficiency. In the context of DC servo motors, the choice between TRZ and BE modeling may also depend on other factors such as the control algorithm used, the requirements for speed and precision, and the specifics of the motor's behavior.

B. Discretized Model of DC Servomotor

Using the BE discretization technique, 7 and 8 can be written as,

$$i_a(k) = i_a(k-1) - \frac{hR_a}{L_a}i_a(k) - \frac{hK_b}{L_a}w_m(k) + \frac{h}{L_a}E_a(k) \tag{12}$$

$$w_m(k) = w_m(k-1) + \frac{hK_t}{J_m} i_a(k) - \frac{hD_m}{J_m} w_m(k) \tag{13}$$

Which can be written in matrix form as,

$$\begin{bmatrix} 1 + \frac{hR_a}{L_a} & \frac{hK_b}{L_a} \\ -\frac{hK_t}{J_m} & 1 + \frac{hD_m}{J_m} \end{bmatrix} \begin{bmatrix} i_a(k) \\ w_m(k) \end{bmatrix} = \begin{bmatrix} i_a(k-1) + \frac{h}{L_a} E_a(k) \\ w_m(k-1) \end{bmatrix} \tag{14}$$

Equation 14 is of $Ax = b$ form where A is the system matrix, x is the variables $i_a(k)$ and $w_m(k)$, which need to be determined, and b is the vector that depends upon the input voltage and the previous sample stored values. Here, A depends only upon the system parameters, and thus remains the same in all samples, whereas b, needs to be determined for each sample. Similarly, using the TRZ discretization technique, 7 and 8 can be written as,

$$i_a(k) = i_a(k-1) + \frac{h}{2}[f_1(k) + f_1(k-1)] \tag{15}$$

where $f_1(k) = -\frac{R_a}{L_a} i_a(k) - \frac{K_b}{L_a} w_m(k) + \frac{1}{L_a} E_a(k)$

$$w_m(k) = w_m(k-1) + \frac{h}{2}[f_2(k) + f_2(k-1)] \tag{16}$$

where $f_2(k) = \frac{K_t}{J_m} i_a(k) - \frac{D_m}{J_m} w_m(k)$ Equations 15 and 16 can be written in matrix form as,

$$\begin{bmatrix} 1 + \frac{hR_a}{2L_a} & \frac{hK_b}{2L_a} \\ -\frac{hK_t}{2J_m} & 1 + \frac{hD_m}{2J_m} \end{bmatrix} x = \begin{bmatrix} i_a(k-1) + \frac{h}{2L_a} E_a(k) + \frac{h}{2} f_1(k-1) \\ w_m(k-1) + \frac{h}{2} f_2(k-1) \end{bmatrix} \tag{17}$$

And by solving 17, which is again of the form $Ax = b$, the variables in x, $i_a(k)$ and $w_m(k)$ can be determined.

Thus, at every sample, k, armature current and angular velocity can be determined from their previous sample value with the present sample voltage E_a.

DC motor modeling program is written in C using both BE and TRZ methods, and it is observed that the TRZ method gives good accuracy as compared to the BE method. That is because the TRZ method is a second-order method thus the local error per sample is proportional to the square of the step size. And can have better stability and convergence properties for stiff differential equations, which is the case with DC servomotor. The stiff mechanical system in DC servomotors ensures that the motor responds quickly and accurately to changes in the input command, and is thus most suitable for tracking and positioning applications.

The control of DC servomotors often involves feedback mechanisms, such as position or speed feedback, to achieve the desired performance characteristics. In this paper, the speed of the DC servomotor is observed and controlled by connecting a properly tuned Proportional-Integral-Derivative (PID) controller in the loop. The motor's actual speed is compared with the reference speed, and the error is fed to the PID controller model. The output of the PID controller becomes the DC input to the DC servomotor.

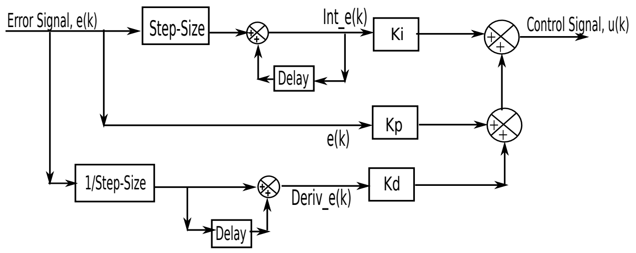

Figure 2. Block diagram of Discretized PID controller

C. Discretized Model of PID Controller

The PID controller output can be expressed as

$$c(t) = K_p e(t) + K_i \int e(t) + K_d \frac{de(t)}{dt} \tag{18}$$

Where $e(t)$ is the error in speed and K_p, K_i, and K_d are the proportional, integral, and derivative constants respectively. It can be discretized as,

$$c(k) = K_p e(k) + K_i Int\_e(k) + K_d Deriv\_e(k) \tag{19}$$

Where at each sample, k, $Int\_e(k)$ represents the integral of the error and $Deriv\_e(k)$ is the derivative of the error. Using BE method, $Int\_e(k)$ can be written as,

$$Int\_e(k) = Int\_e(k-1) + he(k) \tag{20}$$

similarly, derivative of the error can be discretized as,

$$Deriv\_e(k) = \frac{1}{h}(e(k) - e(k-1)) \tag{21}$$

where $(k-1)$ is the previous sample and h is the step-size.

PID controller implementation can be represented as shown in Fig. 2. Here, the error signal $e(k)$ is the input to the PID controller and the control signal $u(k)$ is the output. As shown in the figure, three parallel paths are available, top one is for integral action, in which $Int\_e(k)$ is fed back through a delay block, thus it becomes $Int\_e(k-1)$ and then added with the product of step size, h with $e(k)$. Thus, the equation 20 is implemented in the top path for $Int\_e(k)$, which is then multiplied with K_i to realize $K_i Int\_e(k)$. Similarly, the proportional controller $K_p e(k)$ is realized in the middle path from $e(k)$ and the differential controller $K_d Deriv\_e(k)$ in the third path. Equation 21 is used for the implementation of $Deriv\_e(k)$, in which $e(k-1)$ is generated using the delay block.

The output of the PID controller becomes the input to the DC servomotor for the speed control. The overall block diagram for the speed control of the DC servomotor is shown in Fig. 3. It shows that the actual speed output of the DC servomotor is compared with the reference value and the error is fed to the PID controller to generate the control signal for the motor.

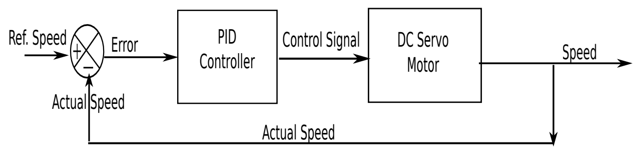

Figure 3. Block diagram for the closed loop speed control of DC servomotor with PID controller

3. Real Time Computation Algorithm

The DC servomotor modeling using both BE and TRZ is coded using the *C* program. The total required simulation time is divided into multiple samples by considering a suitable step-size, *h*.

The computations are performed at every sample, to get the armature current and speed as output by taking the armature voltage as input. The algorithm can be represented as a flow chart as shown in Fig. 4, which can be described as follows.

- Formulate the system matrix, A, using 14 or 17, where 14 is for BE and 17 for TRZ.
- Read the armature voltage,E_a, at current sample, and the armature current, i_a, and angular velocity w_m from the previous sample.
- Formulate the vector b using the above mentioned values.
- Use LU decomposition technique, in which the matrix A is decomposed to LU where L is the unit lower triangular matrix and U is the upper triangular matrix.
- Apply the forward substitution technique and then the backward substitution technique to determine the armature current and angular velocity.
- Store the armature current and angular velocity at the current sample, for the next time-step.
- Determine the speed. Make available the armature current and speed at the output.

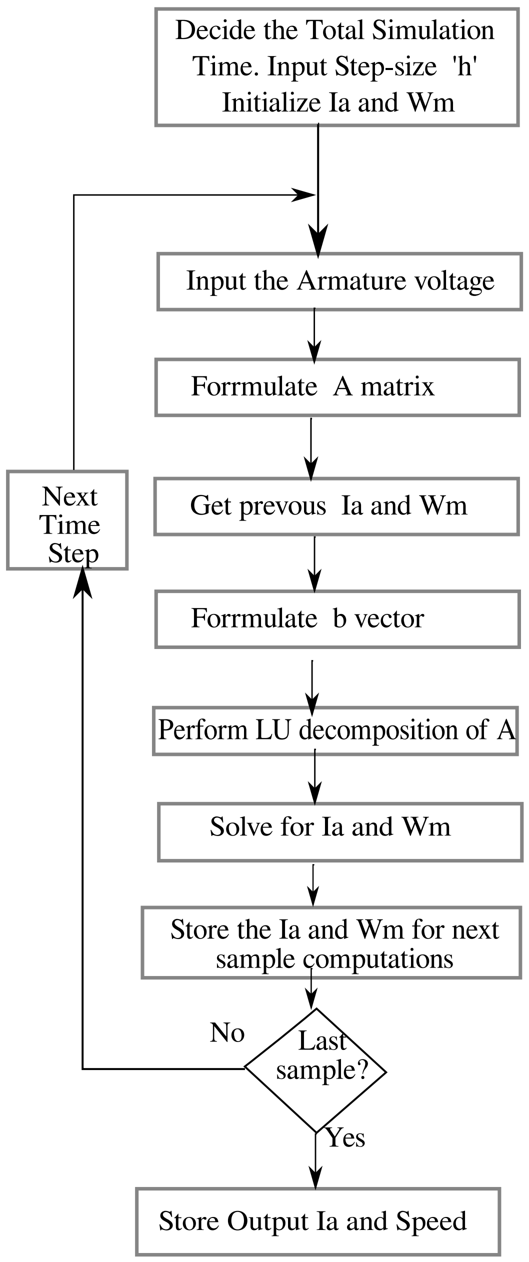
Decide the Total Simulation Time. Input Step-size 'h' Initialize Ia and Wm
Input the Armature voltage
Forrmulate A matrix
Get prevous Ia and Wm
Forrmulate b vector
Perform LU decomposition of A
Solve for Ia and Wm
Store the Ia and Wm for next sample computations
Last sample?
No
Yes
Next Time Step
Store Output Ia and Speed

Figure 4. Flow chart for the computational algorithm

4. Implementation on FPGA

The DC Servomotor speed control is implemented on Xilinx's PYNQ FPGA board using Vivado tools. PYNQ board features Programmable Logic (PL) with an ARM Cortex-A9, 666 MHz Processing System (PS) [40]. It is decided to implement the PID controller and the DC servomotor on the PL and the remaining on the PS. The flow chart for the implementation on FPGA using Vivado tools is shown in Fig. 5 and its detailed description is in the following paragraphs.

The DC Servomotor real-time modeling program is written in *C* code, as explained in section 3, converted to HDL, and corresponding intellectual property (IP) is generated using the Vivado HLS tool [30]. Similarly, *C* code is written for the discrete PID controller implementation, using 19 to 21, which is converted to HDL and then IP using the HLS tool. Here, both DC servomotor IP and PID controller IP are instantiated in the PL using Vivado tools for the closed loop speed control, as shown in Fig. 6.

In this design, error signals are generated in the PS by comparing the input reference speed with the actual speed from the DC servomotor IP. The PID controller IP in PL takes an error as input and generates the control signal for the DC servomotor IP. The DC servomotor IP takes this control signal as input directly from the PID controller in PL

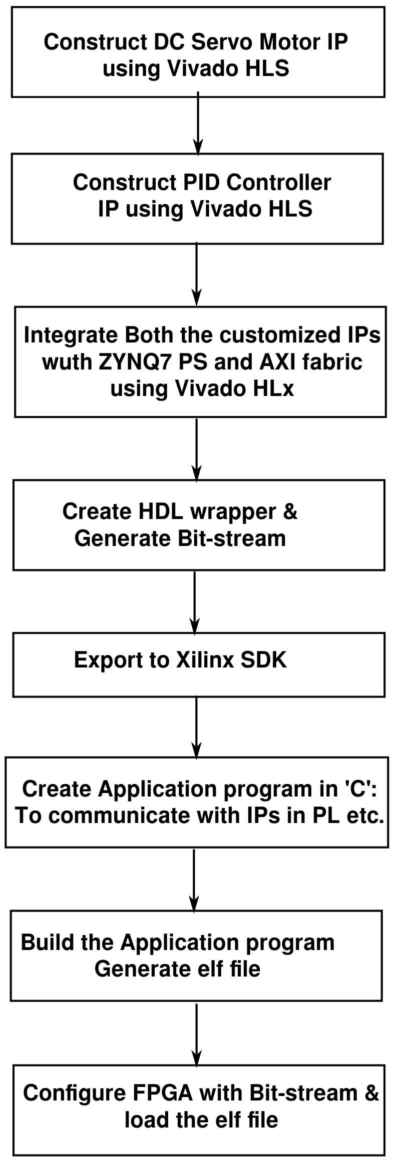
Construct DC Servo Motor IP using Vivado HLS
Construct PID Controller IP using Vivado HLS
Integrate Both the customized IPs wuth ZYNQ7 PS and AXI fabric using Vivado HLx
Create HDL wrapper & Generate Bit-stream
Export to Xilinx SDK
Create Application program in 'C': To communicate with IPs in PL etc.
Build the Application program Generate elf file
Configure FPGA with Bit-stream & load the elf file

Figure 5. Flow chart for the implementation on FPGA, using Vivadoo tools, for the closed loop speed control with PID controller.

through the AXI4-Stream interface (axis). It transfers data in a sequential streaming manner, and thus AXI4-Stream Data FIFO is also included, which provides a buffer between AXI4-Stream data master and slave in the design. Then the speed output of the DC servomotor is given back to the PS for calculating the error.

Since the input port of PID control IP and output port of DC servomotor IP need to be communicated with the PS, AXI4-Lite slave interface (s_axilite) is used during the IP design, which will allow the PS to access the respective port in the IP through the memory-mapped instructions. However as mentioned before, the AXI4-Stream interface (axis) is used for the outport port of PID and input port of

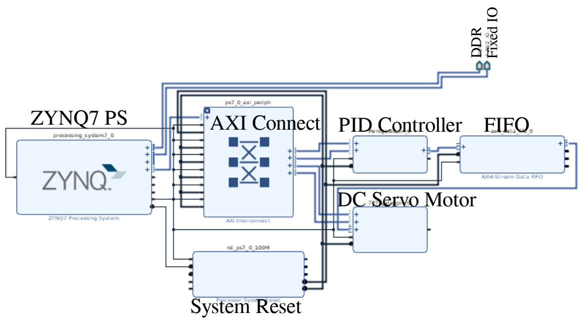

Figure 6. Block Design using Vivado HLx for the closed loop speed control of DC servomotor with PID controller

DC servomotor IPs. The main advantage of this streaming implementation is that it reduces memory-mapped communications and thus reduces the communication latency. The top-level function code for the PID controller IP, with the interface mentioned, is shown below.

```
void PID_IP(float error, float CS)
{
    #pragma HLS INTERFACE s_axilite port=error
    #pragma HLS INTERFACE axi port=CS
    #pragma HLS INTERFACE s_axilite port=return
```

Similarly, in the case of DC servomotor IP, the top-level function code, with the interface mentioned, are

```
void DC_SM_IP(float CS, float Speed, float Ia)
{
    #pragma HLS INTERFACE axi port=CS
    #pragma HLS INTERFACE s_axilite port=Speed
    #pragma HLS INTERFACE s_axilite port=Ia
    #pragma HLS INTERFACE s_axilite port=return
```

As shown in Fig. 6, both of these customized IPs, called as *PID Controller* and *DC Servo Motor*, are integrated with other hardware libraries such as ZYNQ-PS, AXI-Connect, FIFO, etc. available in the Vivado HLx environment to implement the overall hardware system in FPGA.

Then the HDL wrapper is created for the designed system and generates the bit-stream for the FPGA, after the synthesis and implementation on the specified FPGA hardware. Then exports the design to Xilinx's Software Development Kit (SDK) to develop the application program in the on-chip processor. The application program in *C* includes the memory read and write operations and the input-output operation for the communication between the PS and the PL [32].

In our implementation, at every sample, the generated error signals are passed into the PID_IP in PL using the memory-mapped instructions. The instruction Xil_Out32 (u32Addr, u32Value) performs an output operation by writing the 32-bit error value to the address specified. Similarly, 32-bit servomotor outputs, speed, and Ia are read from the address specified using the instruction Xil_In32 (u32Addr). The snippets of our application program code for writing into and reading from the implemented hardware with the signal 1 for starting the communication are shown below:

```
Xil_Out32(PID_IP_BASEADDR+0x10, error);
Xil_Out32(PID_IP_BASEADDR+0x00,1);
while(0==(2 & Xil_In32(DC_SM_IP_BASEADDR+0x00)));
Speed=Xil_In32(DC_SM_IP_BASEADDR+0x20);
Ia=Xil_In32(DC_SM_IP_BASEADDR+0x24);
```

Here, $0x10$, $0x20$, etc. are the offset addresses created during the IP generation in Vivado HLS, which is available at SDK with the exported design. The $0x10$ offset address is for the handshaking signal between PL and PS. Signal 1 indicates the *start* operation for the PL and 2 indicates the output is *ready* by the PL. Once it is ready, the output can be read from the respective address. red Then it is either stored in the off-chip memory available on the PS side of the FPGA, called DDR-DRAM, or made available at the output pins. Now the next iteration starts and it continues till reaching the final desired time.

The above-mentioned application program needs to be built to generate the elf file. Configure the FPGA board with the generated bit-stream and then load the elf file for running the overall system.

5. Results and Discussions

The DC Servomotor is modeled using *C* code with both BE and TRZ methods and the performance is compared with the TF model in MATLAB. The following parameters [9][10] are considered for the modeling of DC Servomotor. $R_a = 2.45$, $L_a = 0.035H$, $K_b = K_t = 1.2\ volt/(rad/sec)$, $J_m = 0.022\ Kg-m^2/rad$, $Dm = 0.5\times10^{-3}N-m/(rad/sec)$.

Total time of 0.25 sec is considered as the simulation time for both BE and TRZ methods using *C* program and for TF in MATLAB. Step-size, *h*, is varied from 20 msec to 0.6 µsec, compared and analysed the performance of BE and TRZ methods with the TF model. The speed at 0.12 sec for all three models with a change in step-size is shown in Table I.

It is clear from the table that, when the step-size is 20 msec, the speed at 0.12 sec corresponds to BE is much deviated from the transfer function model. However, the TRZ method gives a more accurate result as compared to the BE method. When the step-size decreases, the accuracy of both the discretized models improves. At a step-size of 100 µsec, the TRZ result becomes the same as that of TF and at 0.6 µsec step-size, the result corresponds to the BE method also matches with the TF model.

It indicates that being a second order approximation, the TRZ model is more accurate as compared to the BE method and the accuracy increases with the reduction in step-size. However, for a real-time implementation, all communications between the input-output devices with the model, and all the computations in the model need to be completed in the selected step-size. So need to be compromised for the selection of the step-size, and thus second order TRZ method is more advantageous as compared to the BE method.

TABLE I. MEASURED SPEED AT 0.12 SEC FOR ALL THE THREE MODELS FOR VARIOUS STEP-SIZE.

Step-size	Speed Measured at 0.12 sec		
	BE method	TRZ method	Transfer Function
20 msec	1568.1	1613.9	1611.4
10 msec	1586.5	1613.2	1611.4
5 msec	1598.1	1612.3	1611.4
2 msec	1605.8	1611.7	1611.4
100 µsec	1611.3	1611.4	1611.4
20 µsec	1611.3	1611.4	1611.4
6 µsec	1611.3	1611.4	1611.4
0.6 µsec	1611.3	1611.4	1611.4

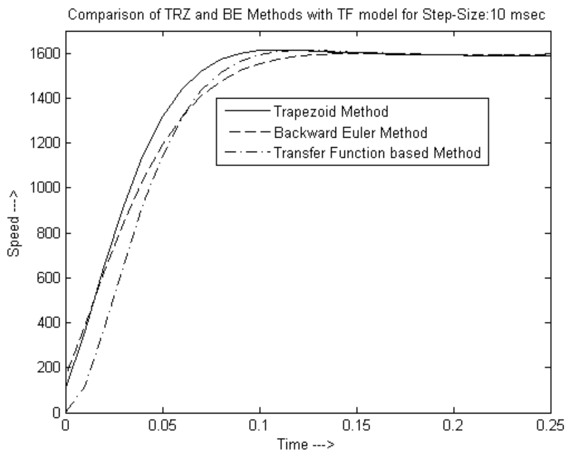

Figure 7. Comparison of TRZ and BE methods with TF for a step-size of 10 msec.

Dynamic response for all three models are plotted in Fig. 7 to 9 with different step-size. It is clear from Fig. 7 that, with a step-size of 10 msec, initial dynamics with both BE and TRZ experiences considerable deviations from that of the TF model, but, TRZ follows more closely to the TF model, as compared to BE.

Fig. 8 shows that, when the step-size is decreased to 5msec, the error is reduced considerably, however, the TRZ result is still nearer to the TF model than the BE.

Now by reducing the step-size further to 6 µsec, both the BE and TRZ become accurate, which is clear from Fig. 9. Only a small deviation is present in the dynamics, which can be seen only by zooming the initial portion, and is shown as a separate rectangle along with the normal view in Fig. 9.

Discrete PID controller is tuned using genetic algorithm details of which are available in paper [10]. With the

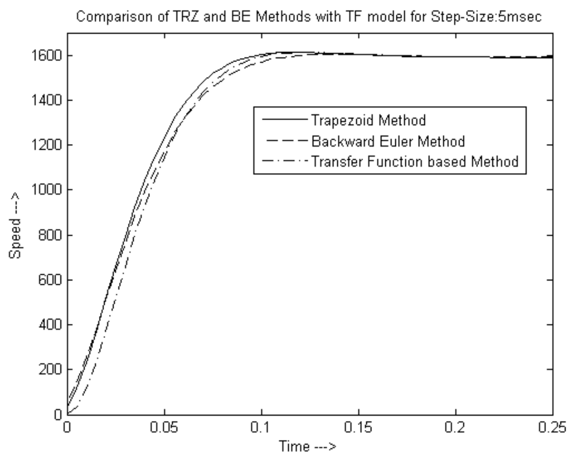

Figure 8. Comparison of TRZ and BE methods with TF for a step-size of 5 msec.

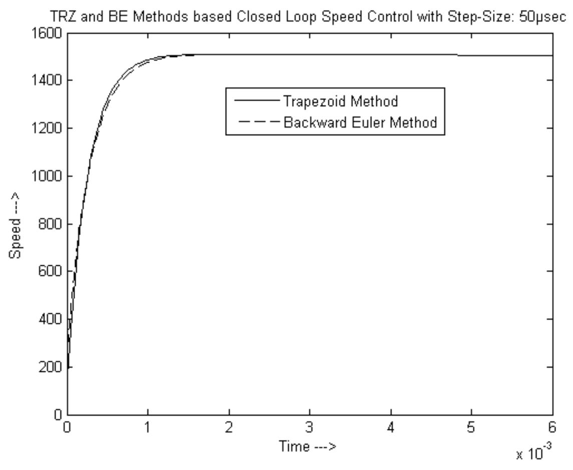

Figure 10. Comparison of TRZ and BE methods for the closed loop speed control with a step-size of 50 µsec.

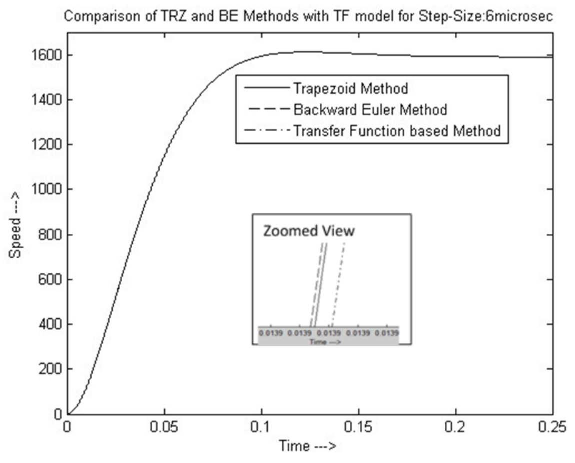

Figure 9. Comparison of TRZ and BE methods with TF for a step-size of 6 µsec.

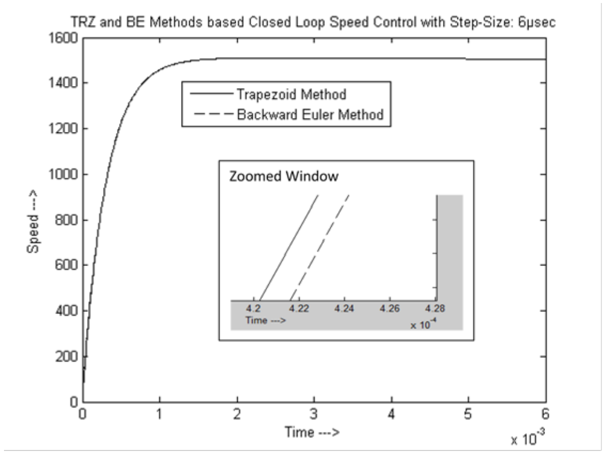

Figure 11. Comparison of TRZ and BE methods for the closed loop speed control with a step-size of 6 µsec

selected values as, K_p=20.875, K_i=0.2138, and K_d=0.2195, the discrete PID controller is modeled using the *C* program and connected in a closed loop with the DC servomotor model for its speed control. The reference speed considered for the analysis is 1500 rad/sec. In this case also, compared the performance of both BE and TRZ modelled DC servomotor and the observations are shown in Fig. 10 and 11, with the step-size varying from 50 µsec to 6 µsec.

Fig. 10 and 11 clearly show that, similar to the previous cases, for higher step-size, TRZ gives better performance. However, when the step-size is reduced, BE is also on par with the TRZ. This shows that TRZ is a better method of discretization, especially with a larger step-size.

Also, the performance is analyzed by changing the reference speed from 1500 rad/sec to 1600 rad/sec at 2.5 msec, and the corresponding plot is shown in Fig. 12.

Fig. 10 and 11 clearly show that with less than 2 msec both BE and TRZ models can achieve the reference speed. A similar conclusion can be inferred from Fig. 12, also with a change in speed.

This analysis shows that by including the discrete PID controller with proper tuning, in the loop with the FPGA based DC servomotor, as expected, a faster dynamic with zero steady state error could be achieved. This strategy can be used for testing and development of various control algorithms for DC servomotor, for diverse applications, without involving the physical motors of numerous possible specifications.

6. Conclusion

The Trapezoidal (TRZ) and the Backward Euler (BE) methods are two implicit methods commonly used for time discretization in the numerical solution of ordinary differential equations (ODE). The relative accuracy of these methods can vary depending on the specific ODE being

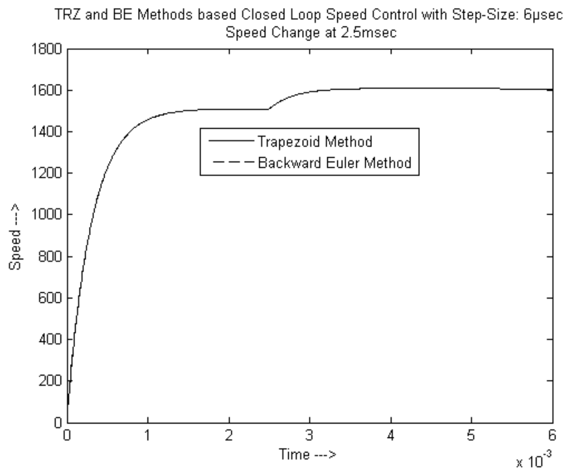

Figure 12. Tracking performance of both TRZ and BE method with speed changed from 1500 rad/sec to 1600 rad/sec at 2.5msec

solved, the step-size chosen, and other numerical considerations. The uniqueness of our paper lies in the application of 'C' for modeling a DC servomotor, employing both BE and TRZ discretization techniques. This paper compares these two discretization methods for the modeling of DC servomotor implemented on FPGA with that of the widely used transfer function method using MATLAB. Step-size is changed from 20msec to 0.6 µsec and observed that the TRZ method gives more accurate results as compared to BE method, especially with a larger step-size. However, shows that the designed FPGA based models of DC servomotor are as accurate as the TF model, and thus can be used for further design purposes.

Speed control performance of the DC servomotor is also analyzed by connecting the models of the DC servomotor in a closed loop with a properly tuned discrete PID controller for a step-size variation from 50 µsec to 6µsec. In this case also, with 50µsec, TRZ was superior, however, at 6µsec, the performance of BE is well matched with the TRZ. This analysis explains in detail the effect of step-size on discretization techniques which will help us to choose the method and the step-size.

By judiciously choosing a step-size and discretization method, an FPGA-based model of a DC servomotor can be developed. The reconfigurable nature of FPGA ensures that the same hardware model can be employed for various applications of DC servomotors, accommodating different specifications. This model proves to be highly effective for designing and testing robust and optimized control systems without the need for physical systems.

The incorporation of the PID controller in the loop analysis further contributes to the seamless integration of FPGA prototypes into control loops. This integration not only enhances the capability to design and advance sophisticated control algorithms for diverse applications but also evaluates the feasibility of the Hardware-in-the-Loop (HIL) technique.

References

[1] Baballe Muhammad, Bello Mukhtar, Abdullahi Umar Abubaka, "A Look at the Different Types of Servo Motors and Their Applications," Global Journal of Research in Engineering & Computer Sciences,vol.2, no. 3, pp. 1-6, 2022.

[2] Blanger, P. Venne, and J. N. Paquin, "The what, where and why of real-time simulation," in Proc. IEEE PES General Meeting, Minneapolis, MN, USA, Jul. 25–29, 2010.

[3] OPAL-RT Technologies, "Real-time solutions for every industry." [Online]. Available: https: //www.opal-rt.com/

[4] X. Guillad et al., "Applications of real-time simulation technologies in power and energy systems," IEEE Power Energy Technol. Syst. J., vol. 2, no. 3, pp. 103–115, Sep. 2015

[5] Liu, Jianying & Zhang, Pengju & Wang, Fei. (2009). "Real-Time DC Servo Motor Position Control by PID Controllers Using Labview," 1. 206 - 209. 10.1109/IHMSC.2009.59.

[6] A. Rai, D. K. Das and M. M. Lotha, "LabVIEW Platform based Real-time Speed Control of a DC Servo Motor With Fuzzy-PI Controller," 2019 International Conference on Electrical, Electronics and Computer Engineering (UPCON), Aligarh, India, 2019, pp. 1-4, doi: 10.1109/UPCON47278.2019.8980036.

[7] Zainab B. Abdullah, Salam Waley Shneen, Hashmia S. Dakheel, "Simulation Model of PID Controller for DC Servo Motor at Variable and Constant Speed by Using MATLAB", Journal of Robotics and Control, Volume 4, Issue 1, January 2023.

[8] N.M. Zakaria and A. O. Elnady, "Implementation of Position Control Servo DC Motor with PID Controller to Humanoid Robot Arm," 5th International Undergraduate Research Conference, Military Technical College, Cairo, Egypt, Aug 9th –Aug 12st, 2021.

[9] T. Hadish, F. Adam, M. Israel, S. Adonay, A. Zebib, "Construction of Mathematical Model of DC Servo Motor Mechanism with PID controller for Electric Wheel Chair Arrangement," Journal of Electronics and Informatics, 3. 49-60, 2021.

[10] Bindu R., Namboothiripad M. K., "Tuning of PID controller for DC servo motor using genetic algorithm", International Journal of Emerging Technology and Advanced Engineering, Volume 2, Issue 3, March 2012.

[11] Neenu Thomas, Dr. P. Poongodi, "Position Control of DC Motor Using Genetic Algorithm Based PID Controller," Proceedings of the World Congress on Engineering 2009 Vol II, WCE 2009, July, 2009.

[12] A. M. Alsayed, E. K. Elsayed, "Optimize Position Control of DC Servo Motor using PID Controller Tuning with Krill Herd algorithm," International Journal of Engineering and Information Systems (IJEAIS),vol. 4, no. 12, pp. 141-147, 2020.

[13] A. M. Abba, T. Karataev, S. Thomas and A. M. Ali, "Optimal PID Controller Tuning for DC Motor Speed Control Using Smell Agent Optimization Algorithm," FUOYE Journal of Engineering and Technology, vol. 7, no.1, pp.23-27, 2022.

[14] R. Kristiyono and W. Wiyono, "Autotuning Fuzzy PID Controller

for Speed Control of BLDC Motor," Journal of Robotics and Control (JRC), vol. 2, no. 5, pp. 400-407, 2021.

[15] H. S. Dakheel, Z. B. Abdullah, N. S.Jasim, S.W. Shneen, "Simulation model of ANN and PID controller for direct current servo motor by using Matlab/Simulink," Telecommunication Computing Electronics and Control , v ol. 20, no. 4, pp. 922-932, 2022.

[16] E. H. Abdelhameed, T.H. Mohamed and G.El-saady, "Design of Hybrid Fuzzy and Position-Velocity Controller for Precise Positioning of a Servo System," International Journal of Applied Energy Systems, vol. 2, no. 2,pp. 111-115, 2022.

[17] Ashna Batool, Noor ul Ain, Arslan Ahmed Amin, Muhammad Adnan, Muhammad Hamza Shahbaz, "A comparative study of DC servo motor parameter estimation using various techniques," Automatika, 63:2, 303-312, Taylor & Francis, 2022.

[18] Kumar, A., Goswami, M. "Performance comparison of instrument automation pipelines using different programming languages," Sci Rep 13, 18579 (2023).

[19] Emertxe, "Why is C the most preferred language for embedded systems?," Sep 21, 2017, https://www.emertxe.com/c-programming/

[20] Murat Sahin, " Designing MPC algorithms for velocity control of brushed DC motor and verification with SIL tests," Automatika 64:3, pages 399-407, Taylor Francis, 2023.

[21] Ruiz-Rosero, Juan, Gustavo Ramirez-Gonzalez, and Rahul Khanna, "Field Programmable Gate Array Applications—A Scientometric Review," Computation 7, no. 4: 63, 2019.

[22] Sadrozinski H.F.W., Wu. J., "Applications of Field-Programmable Gate Arrays in Scientific Research," 1st ed.; Taylor & Francis, Inc., Bristol, PA, USA, 2010.

[23] Rajne, P.A. and Venkataramanan Ramanarayanan, "Programming an FPGA to emulate the dynamics of DC machines," 2006, India International Conference on Power Electronics (2006): 120-124.

[24] Matar, Mahmoud and Reza Iravani, "Massively Parallel Implementation of AC Machine Models for FPGA-Based Real-Time Simulation of Electromagnetic Transients," IEEE Transactions on Power Delivery 26 (2011): 830-840.

[25] V. Ramakrishnan, Nalamwar Sanchit Gopal, R. Ashok and S. Moorthi, "FPGA based DC servo motor control for remote replication of movements of a surgical arm," TENCON 2011 - 2011 IEEE Region 10 Conference, Bali, Indonesia, 2011, pp. 671-675, doi: 10.1109/TENCON.2011.6129192.

[26] Y.J. Zhou, T.X. Mei, "FPGA Based Real Time Simulation for Electrical Machines," IFAC Proceedings Volumes,Volume 38, Issue 1, 2005, Pages 256-261.

[27] AsicNorth, "ASIC vs. FPGA: What to Consider for Your Next Design Project," web document available at https://www.asicnorth.com/blog/asic-vs-fpga-difference/

[28] Sova, V., Grepl, R. (2014), "Hardware in the Loop Simulation Model of BLDC Motor Taking Advantage of FPGA and CPU Simultaneous Implementation," In: Březina, T., Jabloński, R. (eds) Mechatronics 2013. Springer. doi:10.1007/978-3-319-02294-9_18

[29] Tavana, Nariman Roshandel and Venkata R. Dinavahi, "A General Framework for FPGA-Based Real-Time Emulation of Electrical Machines for HIL Applications," IEEE Transactions on Industrial Electronics 62 (2015): 2041-2053.

[30] M. K. Namboothiripad, M. J. Datar, M. C. Chandorkar, and S. B. Patkar, "FPGA accelerator for real-time emulation of power electronic systems using multiport decomposition," inProc. Nat. Power Electron. Conf., 2019, pp. 1–6.

[31] T. Ould-Bachir, H. F. Blanchette, and K. Al-Haddad, "A network tearing technique for FPGA-based real-time simulation of power converters," IEEE Trans. Ind. Electron., vol. 62, no. 6, pp. 3409–3418, Jun. 2015.

[32] M. K. Namboothiripad, M. J. Datar, M. C. Chandorkar and S. B. Patkar, "Accelerator for Real-Time Emulation of Modular-Multilevel-Converter Using FPGA," IEEE 21st Workshop on Control and Modeling for Power Electronics (COMPEL), Aalborg, Denmark, 2020, pp. 1-7.

[33] M. B. Patil, V. Ramanarayanan, V.T. Ranganathan, "Simulation of Power Electronic Circuits," Narosa series in power and energy systems.

[34] Tihamér, Ádám & Dadvandipour, Samad & Futás, József, "Influence of discretization method on the digital control system performance," Acta Montanistica Slovaca. Vol. 8. No. 4 , pp-197-200. December 2003.

[35] Vatansever, Fahri & Hatun, Metin, "s-to-z Transformation Tool for Discretization," Gazi Üniversitesi Fen Bilimleri Dergisi Part C Tasarım ve Teknoloji. 9. 773 - 784, 2021.

[36] B.M. Joshi and M.C. Chandorkar, "Time Discretization Issues in Induction Machine Model Solving for Real-time Applications," conf. rec., IEEE Electric Machines and Drives Conference, 15-18 May 2011, Niagara Falls, Canada.

[37] M. Comanescu, "Influence of the discretization method on the integration accuracy of observers with continuous feedback," 2011 IEEE International Symposium on Industrial Electronics, Gdansk, Poland, 2011, pp. 625-630, doi: 10.1109/ISIE.2011.5984230.

[38] Xilinx, "Introduction to FPGA Design with Vivado High-Level Synthesis," UG998 (v1.0) July 2, 2013.

[39] Ogata, K. (1995), "Discrete-Time Control Systems," Pearson, New York.

[40] Xilinx, "PYNQ-Z2 Reference Manual v1.0", May 17, 2018.

Mini K. Namboothiripad received the B.Tech. degree in electrical and electronics engineering from Govt. Engineering College Thrissur, University of Calicut, Kerala, India in 1995, and the M.Tech. degree in 2011, and the Ph.D. degree in 2021, both in Electrical Engineering from the Indian Institute of Technology Bombay, Mumbai, India. She has been working as an Assistant Professor with the Department of Electrical Engineering, Fr. C. Rodrigues Institute of Technology, Navi-Mumbai, India, since 2001. Her research interests include FPGA-based Fast Computing, Real-Time Simulation, Mathematical Modelling, and Control of Electrical Systems.